Applied Mathematical Modelling 34 (2010) 314–324

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Optimal methods for batch processing problem with makespan and maximum lateness objectives

M.T. Yazdani Sabouni, F. Jolai *

Department of Industrial Engineering, University College of Engineering, University of Tehran, Tehran, Iran

ARTICLE INFO

Article history:
Received 22 January 2008
Received in revised form 4 April 2009
Accepted 29 April 2009
Available online 6 May 2009

Keywords:
Batch processing machine
Bi-criteria scheduling
Multi-agent model
Makespan
Maximum lateness
Dynamic programming

ABSTRACT

In this paper we consider the problem of scheduling n jobs on a single batch processing machine in which jobs are ordered by two customers. Jobs belonging to different customers are processed based on their individual criteria. The considered criteria are minimizing makespan and maximum lateness. A batching machine is able to process up to b jobs simultaneously. The processing time of each batch is equal to the longest processing time of jobs in the batch. This kind of batch processing is called parallel batch processing. Optimal methods for three cases are developed: unbounded batch capacity, $b > n$, with compatible job groups and bounded batch capacity, $b \leqslant n$, with compatible and non compatible job groups. Each job group represents a different class of customers and the concept of being compatible means that jobs which are ordered by different customers are allowed to be processed in a same batch. We propose an optimal method for the problem with incompatible groups and unbounded batches. About the case when groups are incompatible and bounded batches, our proposed method is considered as optimal when the group with maximum lateness objective has identical processing times. We regard this method, however, as a heuristic when these processing times are different. When groups are compatible and batches are bounded we consider another problem by assuming the same processing times for the group which has the maximum lateness objective and propose an optimal method for this problem.

1. Introduction

Here we consider a bi-objective scheduling problem in which a machine is utilized by two agents. In multi objectives scheduling problems it is usually assumed that all of customers have the same objectives. However, in practice maybe this assumption is not held always. Here instead of having same objectives for all agents (customers), we consider a preferred objective for each agent, and agents are satisfied respect to the objectives they pursue. In other words customers select jobs in order to satisfy their own primary objective and these objectives are not applied for all of the customers. Each objective cares about a group of jobs thus different group of jobs represent different customers. So there are different classes of customers and each class of customers selects its own jobs according to the preferred criterion. In the scope of multi agents scheduling problems little work has done. Agnetis et al. [1] consider a job shop problem with two jobs and with two customers. This work shows how to find none dominated schedules in order to make able the customers to satisfy their personal objectives through optimal and efficient ways. Agnetis et al. [2] consider a multi objectives scheduling problem with two customers. In this model the first customer's objective function is optimized by assuming the constraint that other customer's objective function is bounded. They also assume three classes of objective function: minimizing the maximum of

* Corresponding author.
E-mail address: fjolai@ut.ac.ir (F. Jolai).

0307-904X/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2009.04.007

a regular function, minimizing the number of tardy jobs and minimizing weighted completion time. Baker and Smith [3] study a multi agents scheduling problem that each customer pursues different objectives. In this paper three common objective functions are considered: minimizing the completion time of jobs, minimizing the maximum lateness and minimizing total weighted completion time. They show that this kind of scheduling model with the objectives L_{max} and $\sum w_j C_j$ is NP-complete.

In our work, we consider the parallel batch processing problem to minimize makespan and maximum lateness objectives. A parallel batch process machine can handle some jobs in a batch at the same time. All jobs in this batch have the same starting and ending times and the processing time of this batch is the longest processing time of the jobs in it. We classify the jobs into two groups and each group represents a different customer. In other words each customer has to select his jobs from his own group. In this way each customer has a direct role to minimize his objective function but he indirectly effects on the other customer's criterion. Sometimes due to technical constraint, different groups cannot insert their jobs in a similar batch. We call this case as the incompatible groups. On the other hand, however, there might be some cases that different customers can process their jobs in a same batch and note this case as compatible groups. In the literature of batching problems it is usually assumed that batches have capacity. So if we consider sizes for the jobs, the total sizes of the jobs which a batch contains must not exceed the capacity of the batch. This case is usually called the problem with bounded batches. As far as each job may have a different size, the number of jobs each batch contains may be different. On the other hand if any number of jobs is allowed to be inserted in a batch, we have unbounded batches. In this case batches are not restricted in processing of any number of jobs.

We consider two main categories in this paper which are based on the compatibility or lack of it among the job groups. In the case of incompatibility, we assume batches are bounded and unbounded and also about the problem with compatible groups, we consider bounded batches. An approach is given in He et al. [4] and we can use this approach for the case with compatible groups and unbounded batches with a little modification, so we do not pay attention to this case in details. Hence three different classes of problems are derived and we propose an optimal method for each one. Although, the proposed method for the problem with incompatible job groups and bounded batches when processing times of the group with maximum lateness objective are identical can be considered as optimal, we note there if these jobs have different processing times, this method is taken into account as a heuristic. Other assumptions about the considered problems are that jobs have identical sizes when we assume bounded batches. In the problem with compatible job groups and bounded batches, we as-

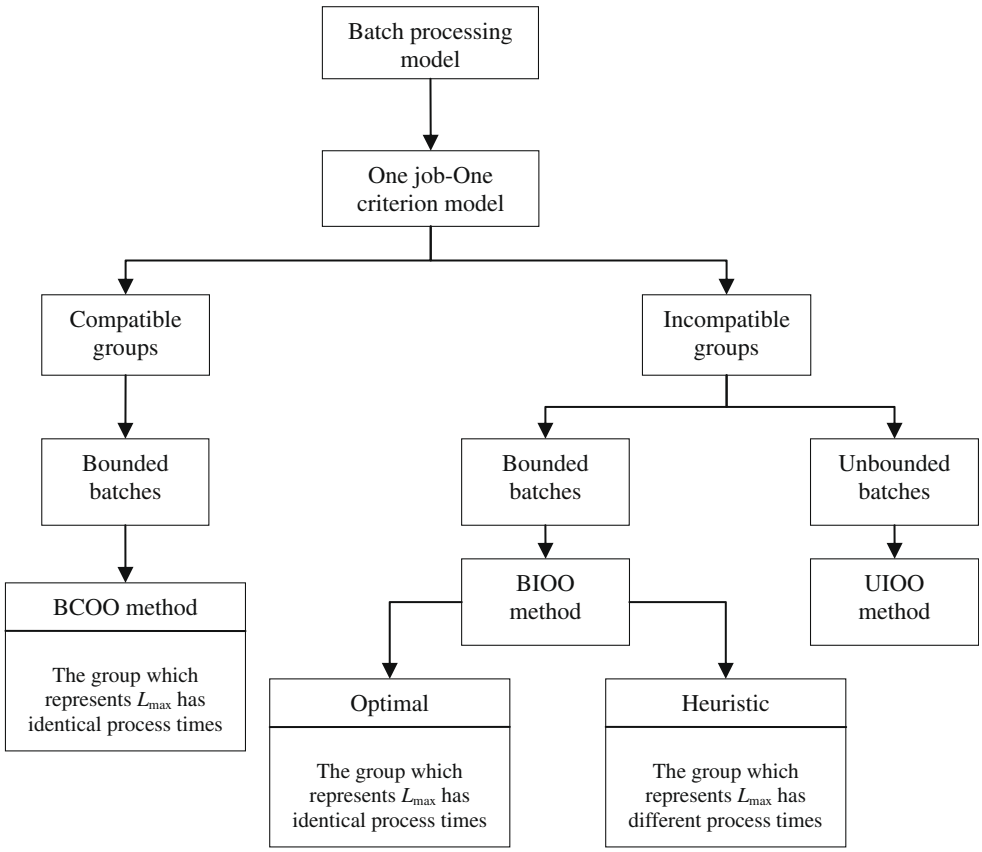

Fig. 1. The proposed methods for the problems which are derived from the One job–One criterion model.

sume that jobs of the group with maximum lateness objective have the same processing times. We note the problems which are considered in our work by:

$1|p\text{-}batch, b > n, IG, mul\text{-}cust|F(C_{max}, L_{max})$, $1|p\text{-}batch, b \leqslant n, IG, mul\text{-}cust|F(C_{max}, L_{max})$ and $1|p\text{-}batch, b \leqslant n, CG, mul\text{-}cust|F(C_{max}, L_{max})$ where *IG* and *CG* imply incompatible and compatible groups, respectively. The overall view of the considered problems with different attributes is shown in Fig. 1. The rest of this paper is outlined as follows:

In Section 2 we give a brief description of the notions which we use throughout our work and since in some parts of this work we apply the Pareto sense, we describe the notion of Pareto optimal solution. The model which has several customers who select their preferred jobs from different job groups is given in Section 3 in three different classes of problems. The computational experiments are given in the Section 4. Finally, Section 5 provides conclusions of the study.

Through the rest of our paper we note the model in which one of customers wants to minimize the makespan criterion and the other one seeks for minimizing the maximum lateness as One job–One criterion model. Regarding to that whether the groups are compatible or incompatible and batches are bounded or unbounded, three diverse problems are considered and a method (an optimal and in a case a heuristic) is proposed for each one.

2. Basic notion and concepts

Assume there are n jobs, we show them by $J_1, J_2, \ldots, J_n$. These jobs have to be processed on a batch process machine by starting from time zero. The job $J_j (j = 1, \ldots, n)$ has processing time p_j and due date d_j. By having a schedule σ, we note the completion time of J_j in σ by $C_j(\sigma)$. Without losing the generality, we assume all processing times and due dates are integral. For the schedule σ, $L_j(\sigma) = C_j(\sigma) - d_j$ and $L_{max}(\sigma) = \text{Max}\{L_j(\sigma)\}(j = 1, \ldots, n)$ are equal to lateness of job J_j and maximum lateness of σ, respectively. For problems with minimizing objective function without jobs release dates, there must be an optimal schedule in which batches are processed continuously from time zero onwards. Hence the schedule σ is a sequence of batches like the form $\sigma = (B_1, B_2, \ldots, B_r)$ such that each batch contains a set of jobs. The processing time of the batch B_l is $p(B_l) = \text{Max}\{p_j\}(J_j \in B_l)$ and its completion time is $C(B_l) = \sum p(B_q)(q = 1, \ldots, l)$. Since all jobs in a batch have the same completion time, the completion time of job $J_j \in B_l$ for $1 \leqslant l \leqslant r$ in schedule σ, is equal to $C_j(\sigma) = C(B_l)$.

2.1. Pareto ranking approach

The common approach for balancing the objectives in a good way is Pareto ranking. The first Pareto ranking method was introduced by Goldberg [5]. In this approach one element in the search space is said Pareto optimal if it is not less than another element in the space according to all objectives. For a minimization problem, for two individuals $U(u(1), u(2), \ldots, u(k))$ and $V(v(1), v(2), \ldots, v(k))$ in which u and v are objective functions, we say that U dominates V iff for all i $u(i) \leqslant v(i)$ and there exists at least one individual i where $u(i) < v(i)$.

3. One job–One criterion model

This model considers two independent classes of customers such that one class looks for the C_{max} criterion and the other class wants to visit the optimum value of L_{max}. The jobs in each class are directly scheduled in order to optimize their relative objective function and also they have an indirect impact on the objective of the other class. By having this model, we consider three problems which are differentiated by the existence of compatibility among job groups and the capacity of batches. So in this section we discuss about three problems and propose optimal methods for them. The optimality of the proposed methods is based on bellman 's principal of optimality in dynamic programming [6].

3.1. The methods for the problems with incompatible job groups

3.1.1. Objective function when groups are incompatible

In this section we assume incompatible job groups. Therefore, jobs from different groups have to be processed in separate batches. Also we study here two problems, one problem has bounded batches and the other has unbounded ones. Also we assume a total weighted function of the objectives C_{max} and L_{max}, and let customers from different classes select their own jobs, but the final decision is taken into account by assigning the jobs into an arrangement which satisfies an average of those self criteria. We show the classes with C_{max} and L_{max} objectives by C_1 and C_2, respectively. Suppose the objective Z_1 is applied for the class C_1 and the objective Z_2 is applied for the class C_2. Then the objective function which considers the objective of C_1 together with the objective of C_2 is like the form $Z = \theta Z_1 + (1 - \theta)Z_2$ where $\theta > 0$ is a weighting factor which helps the objectives to be scaled.

3.1.2. The UIOO method for $1|p\text{-}batch, b > n, IG, mul\text{-}cust|F(C_{max}, L_{max})$[1]

Assume the class C_1 represents the objective C_{max} and the class C_2 considers the objective L_{max}. Also because batches are unbounded, any number of jobs can be inserted in a batch but since job groups are incompatible, the jobs in a same batch

[1] Unbounded Incompatible One job–One criterion.

must be from a similar job group. For this problem we propose a method by which we can find the optimal schedule in order to minimize C_{max} and L_{max} criteria. We give Property 1 in the following which tells us all jobs of the class C_1 should be put in a batch. Then, we provide an algorithm which sequences jobs of the class C_2 in batches optimally. Finally, we describe an approach which finds the best place for putting the batch containing all jobs of the class C_1 between every two consecutive batches of the class C_2.

Property 1. All jobs in the class C_1 must be inserted in a same batch.

Proof. Clearly if we insert all jobs in a batch, the total completion time is the maximum processing time among all jobs. But when we put jobs in more than one batch, the total completion time is definitely larger than the maximum processing time of the jobs. Thus putting all jobs in a batch gives us the minimum total completion time for the class C_1.

As a consequence of Property 1, by considering that batches are unbounded, we can insert all jobs of the class C_1 in a batch and consider this batch as a job with the processing time and the due date which are the maximum processing time and minimum due date of the jobs in it.

In the following we describe Lemma 1 together with Property 2 and use them for providing a primary sequence of jobs of the class C_2. □

Lemma 1. *In order to minimize the L_{max} criterion, there is an optimal schedule for jobs of the class C_2, in which jobs are arranged in SPT order.*

Proof. Consider the optimal schedule $\sigma = (B_1, \ldots, B_\gamma, \ldots, B_q \ldots, B_r)$ such that: for $J_j \in B_q$, $J_k \in B_\gamma (1 \leqslant \gamma \leqslant q \leqslant r)$ we have $p_k > p_j$. Now consider the schedule σ' which is gained by inserting the job J_j from the batch B_q to B_γ. Thus $\sigma' = (B_1, \ldots, B_\gamma \cup \{J_j\}, \ldots, B_q - \{J_j\} \ldots, B_r)$. Since $p_k > p_j$, we have $p(B_q - \{J_j\}) \leqslant p(B_q)$, $p(B_\gamma \cup \{J_j\}) = p(B_\gamma)$ and $L_i(\sigma') \leqslant L_i(\sigma) (i = 1, \ldots, n)$.

Thus $L_{max}(\sigma') \leqslant L_{max}(\sigma)$, $C_{max}(\sigma') \leqslant C_{max}(\sigma)$. Hence the new schedule σ' is also optimal. The finite number of these repetitions provides the optimal schedule. □

Property 2. For Jobs of the class C_2 there is an optimal solution such that the jobs $J_1, J_2, \ldots, J_n$ can be re indexed in the way $p_1 < p_2 < \ldots < p_n$ and $d_1 < d_2 < \ldots < d_n$.

Proof. If there exist two jobs J_i and J_j which $p_i \leqslant p_j$ and $d_i \leqslant d_j$, then it is possible to place the job J_i in the same batch which contains J_j without increasing completion time and hence maximum lateness for the class C_2. □

Table 1
Results for the problem 1|*p-batch*, $b > n$, *IG*, *mul-cust*| $F(C_{max}, L_{max})$ and 50-job instances.

Due date type	θ	CPU time		Objective value	
		Mean	Max	Mean	Max
$n = 50$					
I	0.1	0.001	0.015	101.1	108.9
	0.2	0.002	0.016	105.1	118.1
	0.3	0.002	0.016	104.3	120.9
	0.4	0.002	0.016	108.4	129.2
	0.5	0.002	0.016	107.9	136.5
	0.6	0.002	0.016	129.9	149.4
	0.7	0.002	0.016	113.3	156.4
	0.8	0.002	0.016	121.2	177.6
	0.9	0.002	0.016	115.8	163.2
II	0.1	0.002	0.016	97.1	109.5
	0.2	0.002	0.016	99.2	119.8
	0.3	0.002	0.016	102.4	114.8
	0.4	0.002	0.016	97.5	124.9
	0.5	0.002	0.016	103.2	130.1
	0.6	0.002	0.016	100.8	150.4
	0.7	0.001	0.015	95.2	152.6
	0.8	0.001	0.016	112.2	161.8
	0.9	0.001	0.015	107.9	185.4
III	0.1	0.001	0.015	101.4	108.1
	0.2	0.002	0.016	96.9	113.8
	0.3	0.001	0.015	104.7	120.1
	0.4	0.002	0.016	110.4	128.2
	0.5	0.001	0.015	109.4	140.1
	0.6	0.001	0.016	87.1	149.8
	0.7	0.001	0.015	84.2	152.9
	0.8	0.001	0.015	112.5	166.4
	0.9	0.001	0.015	131.9	179.1

Table 2
Results for the problem 1|*p-batch*, *b* > *n*, *IG*, *mul-cust*| $F(C_{max}, L_{max})$ and 100-job instances.

Due date type	θ	CPU time		Objective value	
		Mean	Max	Mean	Max
$n = 100$					
I	0.1	0.011	0.016	103.7	109.8
	0.2	0.011	0.016	103.5	117.8
	0.3	0.011	0.016	109.5	126.1
	0.4	0.011	0.016	106.5	129.2
	0.5	0.011	0.016	116.1	141.1
	0.6	0.011	0.016	108.7	154.6
	0.7	0.011	0.016	115.7	165.8
	0.8	0.011	0.016	127.1	166.4
	0.9	0.011	0.016	136.1	173.7
II	0.1	0.011	0.016	99.5	108.3
	0.2	0.011	0.016	101.1	118.4
	0.3	0.011	0.016	90.9	120.9
	0.4	0.011	0.016	114.3	134.8
	0.5	0.011	0.016	112.2	141.1
	0.6	0.011	0.016	118.1	150.4
	0.7	0.011	0.016	106.6	151.8
	0.8	0.011	0.016	106.4	166.4
	0.9	0.011	0.016	110.4	167.3
III	0.1	0.012	0.016	100.4	107.6
	0.2	0.012	0.016	99.4	117.2
	0.3	0.012	0.016	106.6	125.5
	0.4	0.012	0.016	109.5	136.6
	0.5	0.012	0.016	101.1	147.1
	0.6	0.012	0.016	103.4	140.4
	0.7	0.012	0.016	97.8	153.7
	0.8	0.012	0.016	112.9	176.8
	0.9	0.012	0.016	97.9	164.7

After providing the primary schedule for entire jobs of the class C_2, we place these jobs in batches by our following proposed algorithm:

Algorithm 1. Assume there are n_1 jobs in the class C_2. Regarding to the Property 2, the first job in each batch has the earliest due date. Hence we can consider the due date of the first job in each batch as the due date of that batch. Assume $F(0) = 0$ as the initialization and it is jth job's turn to be scheduled. Also let $F(J_j)$ be the minimum makespan for $J_1, J_2, \ldots, J_j$ jobs and $L(J_j)$ be the lateness of J_j. For computing the optimal schedule we have the following DP algorithm:

We find $L(J_j)$ for $j = 1, \ldots, n_1$:

$$L(J_j) = \text{Min}\{\text{Max}\{L(J_k), F(J_K) + p_j - d_{k+1}\}\}(0 \leqslant k < j) \tag{1}$$

Suppose the smallest value of $\text{Max}\{L(J_k), F(J_K) + p_j - d_{k+1}\}$ happens in $k = \bar{k}_j$. Then we put $F(J_j) = F(\bar{k}_j) + p_j$. Thus the optimal schedules is gained by keeping the batch $\{J_{\bar{k}_{n_1}} + 1, \ldots, J_{n_1}\}$ as the last batch and the batch $\{J_{\bar{k}_{i+1}}, \ldots, J_{\bar{k}_{n_1}}\}$ as the second last batch and we continue until we get the first batch. Therefore, when the value of $F(J_{n_1})$ is obtained, we move backward to the last $J_{\bar{k}_{n_1}}$ which $F(J_{n_1}) = F(J_{\bar{k}_{n_1}}) + p_{n_1}$. We do it again by starting from $J_{\bar{k}_{n_1}}$ and gain the second last batch. So we continue this fashion until we reach the first batch. The optimal value of L_{max} is obtained by calculating $F(J_{n_1})$ and $L(J_{n_1})$.

Now by having the Property 1 we know that all jobs of the class C_1 have to be assigned in just one batch and also we order the jobs of class C_2 into optimal by using the Algorithm 1.

Respect to the following description, we can find an optimal schedule by finding the best place for the batch containing all jobs of the class C_1 between the batches of the class C_2. Assume we have formed the batch containing all jobs of the class C_1 (name this batch B_{total}) as well as the optimal schedule for jobs of the class C_2 (name this schedule σ). Suppose the maximum lateness among jobs of the class C_2 occurs in the batch B_1. Now we put B_{total} at the beginning of the schedule σ before all of its batches, name this schedule H_1. Notice there is no need to place the batch B_{total} anywhere else before the batch B_1 (except the beginning of σ). That is because the objective of C_1 group or C_{max} would get worse while the objective function of the C_2 group or L_{max} would not change (compare to when B_{total} is positioned at the beginning). The objective function for this schedule is $Z_1 = \theta C_{Btotal} + (1 - \theta)L_{B1}(H_1)$ where $L_{B1}(H_1)$ is the lateness of B_1 in H_1. Then we put the batch B_{total} immediately after B_1. Name this new schedule H_2. Suppose that B_2 be a C_2 batch which attains the maximum lateness among the jobs of C_2. The objective function in this case is $Z_2 = \theta(C_{Btotal} + C_{B1}(\sigma)) + (1 - \theta)L_{B2}(H_2)$ where $C_{B1}(\sigma)$ is the completion time of B_1 in σ. Similarly we cannot assume the place of B_{total} anywhere except immediately after the batch B_1 or immediately after the batch B_2. We put B_{total} after B_2 and name the resulted sequence H_3. We May continue this process until we reach the end of the σ

schedule. Then from all of the possible cases, the best possible place for the batch containing C_1 jobs among C_2 batches is the one which meets the minimum value of the objective function Z.

3.1.3. The BIOO method for 1|p-batch, $b \leqslant n$, IG, mul-cust|$F(C_{max}, L_{max})$[2]

Let the class C_1 and C_2 represent the classes with the objectives $C_{\max}$ and $L_{\max}$, respectively. Because batches are bounded, limited number of jobs is allowed to be inserted in a batch and since job groups are incompatible, the jobs in a same batch must be from similar job group. In this problem we assume all jobs of a same group have the same sizes which are equal to one for all of them. We propose a heuristic method in order to minimize $C_{\max}$ and $L_{\max}$ criteria. However, when jobs of the class C_2 have the same processing times, this method can be used as an optimal method. In the following, by applying the Property 3, we assign jobs of the class C_1 into batches as much as possible. Then we provide an algorithm which sequences jobs of the class C_2 in batches. Finally we find the place of C_1 batches between C_2 batches.

Property 3. Respect to that all jobs of the class C_1 have the same sizes, there is an optimal schedule that allows SPT order for the jobs of the class C_1 and we fill each batch until it will be full completely.

Proof. Since batches are bounded and the jobs of a same group have the same size, we have to insert jobs in batches as much as possible. Because the processing time of a batch is equal to the longest processing time of jobs in it, it is better to move the jobs with large processing time to end of the schedule. Thus for a schedule we start batching from the jobs with small processing times and keep jobs whose processing times are large at the end of the schedule. Therefore, the SPT order arranges jobs of the class C_1 in the optimal sequence. □

Algorithm 2. We suggest the EDD sequence of jobs to be a promising starting point. Assume the class C_2 includes n_1 jobs and the capacity of each batch in this class is b. Considering EDD order in this class, the first job in each batch has the earliest due date and so we can consider it for the batch. For computing the schedule for jobs of C_2, we have the following:

Let $F(J_j)$ be the makespan after assigning the job J_j to the machine. The initialization is $F(0) = 0$ and for $j = 1, \ldots, n_1$ we have the following:

$$L(J_j) = \mathrm{Min}\{\mathrm{Max}\{L(J_k), F(J_K) + p_{\max} - d_{k+1}\}\}\left(0 \leqslant k < j, k+1 \leqslant i \leqslant j, \sum s_i \leqslant b\right) \tag{2}$$

where $p_{\max} = \mathrm{Max}\{p_r\}(k+1 \leqslant r \leqslant j)$

It is seen in this formula that creating batches is done by considering the batch capacity restriction. Since all jobs in this group have the same sizes which are equal to one, the total number of jobs which are allowed to be inserted in a batch is less than b.

We consider the minimum value of Max $\{L(J_k), F(J_K) + p_{\max} - d_{k+1}\}$ occurs in $k = \bar{k}_j$. Then we put $F(J_j) = F(\bar{k}_j) + p_j$. Then the best possible schedule is gained by considering $\{J_{\bar{k}_{n_1}+1}, \ldots, J_{n_1}\}$ as the last batch and $\{J_{\bar{k}_i+1}, \ldots, J_{\bar{k}_{n_1}}\}$ as the second last batch and so on, similarly we have other batches. To gain the final schedule in this class, we have to calculate the values of $F(J_{n_1})$ and $L(J_{n_1})$ and then obtain other batches by backtracking. The complexity of this problem when processing times are different is studied by Brucker et al. [7] and they show that this case is NP-hard. The schedule which is obtained by Algorithm 2 is not optimal. Thus by having different processing times, we can consider this method as a heuristic. We note here that Algorithm 2 provides the optimal schedule when all of processing times in C_2 are similar and equal to p. In this sense the recursive relation (2) is changed into the following relation:

$$L(J_j) = \mathrm{Min}\{\mathrm{Max}\{L(J_k), F(J_K) + p - d_{k+1}\}\}\left(0 \leqslant k < j, k+1 \leqslant i \leqslant j, \sum s_i \leqslant b\right) \tag{3}$$

For this case, similarly, the batching stage is the same to the scenario which is mentioned for the previous problem. When the optimal schedule of the C_1 jobs (name it the schedule σ) is built and also the final schedule (optimal or the heuristic solution) for jobs of the class C_2 is obtained (name it the schedule γ), it is the time to trade the places of the batches of the both schedules σ and γ in a sequence. Similar to the approach which is described in UIOO method, sequentially we find the best position for the batches of σ among the batches of γ.

3.2. The method for the problem with compatible job groups

3.2.1. The BCOO method for 1|p-batch, $b \leqslant n$, CG, mul-cust|$F(C_{max}, L_{max})$[3]

The problem without considering job groups and different agents while batches are unbounded is studied by He et al. [4]. They give a dynamic programming algorithm which is able to find the minimum makespan by restricting maximum lateness to a given $L_{\max}$ value. Their method can be used for the problem with customers and job groups by a little modification in the DP algorithm. Therefore, we neglect the description for 1|*p-batch*, $b > n$, *CG*, *mul-cust*|$F(C_{\max}, L_{\max})$.

[2] Bounded Incompatible One job–One criterion

[3] Bounded Compatible One job–One criterion.

Table 3
Results for the problem 1|*p-batch*, $b > n$, *IG*, *mul-cust*|$F(C_{max}, L_{max})$ and 150-job instances.

Due date type	θ	CPU time		Objective value	
		Mean	Max	Mean	Max
$n = 150$					
I	0.1	0.021	0.032	103.9	109.2
	0.2	0.021	0.032	103.2	112.6
	0.3	0.021	0.047	109.1	126.1
	0.4	0.021	0.047	120.8	135.4
	0.5	0.022	0.046	120.1	141.1
	0.6	0.021	0.047	109.2	148.2
	0.7	0.021	0.047	126.3	166.5
	0.8	0.017	0.032	113.1	166.4
	0.9	0.021	0.047	126.5	185.5
II	0.1	0.022	0.047	101.4	106.1
	0.2	0.021	0.047	104.5	112.1
	0.3	0.021	0.047	98.5	122.5
	0.4	0.022	0.047	102.2	130.4
	0.5	0.022	0.047	104.8	140.1
	0.6	0.022	0.047	123.3	154.1
	0.7	0.022	0.047	108.1	148.3
	0.8	0.021	0.047	123.6	165.6
	0.9	0.021	0.047	110.1	172.9
III	0.1	0.022	0.047	100.7	108.3
	0.2	0.021	0.047	104.3	118.2
	0.3	0.021	0.047	100.6	123.4
	0.4	0.022	0.047	97.8	131.6
	0.5	0.022	0.047	105.6	130.5
	0.6	0.022	0.047	103.6	144.4
	0.7	0.022	0.046	117.4	165.1
	0.8	0.022	0.047	97.8	159.2
	0.9	0.022	0.047	95.7	185.5

Table 4
Results for the problem 1|*p-batch*, $b \leqslant n$, *IG*, *mul-cust*|$F(C_{max}, L_{max})$ and 50-job instances.

Type	θ	Capacity of batches											
		5				10				15			
		CPU time		Objective value		CPU time		Objective value		CPU time		Objective value	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
$n = 50$													
I	0.1	0	0	146.1	153.1	0.001	0.016	114.4	126.1	0.001	0.016	115.1	122.4
	0.2	0.001	0.016	187.8	206.2	0.001	0.015	130.3	148.1	0.001	0.016	95.8	143.4
	0.3	0	0	221.9	269.9	0.001	0.016	72.1	186.7	0.001	0.015	47.1	120.3
	0.4	0.001	0.016	222.4	312.1	0.001	0.015	−44.9	113.2	0	0	−36.6	87.4
	0.5	0.001	0.016	174.7	290.1	0.001	0.016	−112.3	115.1	0.001	0.016	−30.4	111.1
	0.6	0.001	0.016	50.8	129.6	0.001	0.016	−210.5	−4.9	0.001	0.016	−161.6	29.1
	0.7	0	0	21.4	146.1	0.001	0.016	−351.1	−111.9	0.001	0.016	−255.8	46.5
	0.8	0	0	−136.1	−40.1	0.001	0.016	−318.9	−46.4	0.001	0.015	−296.1	−51.4
	0.9	0.001	0.016	−204.3	−78.8	0.001	0.015	−511.6	−157.7	0.001	0.015	−274.4	−25.6
II	0.1	0.001	0.016	139.2	153.6	0.001	0.016	111.1	124.3	0.001	0.016	109.6	118.9
	0.2	0	0	177.8	211.4	0.001	0.015	77.7	141.8	0.001	0.016	71.7	111.2
	0.3	0.001	0.016	188.7	251.4	0	0	−46.1	96.2	0.001	0.015	−70.1	27.1
	0.4	0.001	0.016	81.6	211.4	0.001	0.016	−218.1	−32.1	0	0	−87.5	49.6
	0.5	0.001	0.015	−37.1	109.1	0.001	0.016	−379.5	−129.1	0.001	0.016	−226.3	76.5
	0.6	0.001	0.015	−204.1	10.8	0.001	0.016	−517.4	−60.6	0.001	0.016	−481.8	−245.8
	0.7	0.001	0.016	−312.1	−176.6	0.001	0.016	−617.6	−375.4	0.001	0.016	−499.6	−163.1
	0.8	0.001	0.016	−481.6	−198.8	0	0	−790.7	−408.4	0.001	0.015	−679.3	−354.2
	0.9	0	0	−650.4	−350.3	0	0	−883.6	−570.1	0.001	0.016	−695.2	−198.1
III	0.1	0.001	0.015	139.1	150.1	0.001	0.015	111.6	124.7	0.001	0.016	110.1	126.4
	0.2	0.001	0.016	166.2	209.1	0	0	78.9	143.1	0.001	0.015	33.9	137.1
	0.3	0.001	0.016	178.9	233.4	0.001	0.015	−63.8	69.9	0.001	0.016	−42.4	56.9
	0.4	0.001	0.016	72.6	191.4	0.001	0.016	−219.1	−94.1	0.001	0.015	−213.3	11.8
	0.5	0.001	0.015	−107.5	40.1	0	0	−405.8	−198.5	0	0	−307.5	−60.1
	0.6	0	0	−192.2	−55.1	0.001	0.016	−477.2	−155.8	0.001	0.016	−356.6	−31.1
	0.7	0.001	0.015	−369.5	−172.7	0.001	0.016	−619.8	−319.3	0.001	0.015	−419.6	−88.5
	0.8	0	0	−536.6	−178.1	0.001	0.015	−791.4	−402.4	0	0	−535.4	−170.4
	0.9	0.001	0.015	−697.1	−170.5	0.001	0.015	−812.1	−287.8	0.001	0.016	−669.1	−193.1

Consider the classes C_1 and C_2 have the objectives C_{max} and L_{max}, respectively. Also it is assumed here that batches are bounded and due to compatibility between job groups, jobs in a same batch can be selected from different job groups. For this problem we assume that all jobs in the class C_2 have the same processing times while jobs in the class C_1 may have different processing times. Also all jobs of the two groups have the same sizes which we assume are equal to one. At first two properties are given and regarding to them we sequence jobs of the class C_1 and C_2.

Property 4. Regarding to that jobs of the class C_1 lack due dates or $J_j \in C_1$, $d_i = +\infty$, there is an optimal sequence in which the jobs in this group are sequenced in SPT. The SPT order reduces the C_{max} objective in the group C_1 and it has the role in decreasing L_{max} for the class C_2 indirectly.

Property 5. Since jobs of the class C_2 have the same processing times, in order to minimize the L_{max} criterion we have to sequence these jobs in EDD.

Here we propose a method which solves this problem optimally. Assume the groups C_1 and C_2 contain n_1 and n_2 jobs, respectively, therefore we have to process $n = n_1 + n_2$ jobs in batches totally. At first regarding to the Property 4, we arrange jobs of the class C_1 in SPT and also respect to Property 5, jobs of the class C_2 are arranged in EDD order. In the following we propose the BCOO method to tackle the problem.

Assume $F(J_{j-1})$ is the minimum completion time for the batches containing the jobs $J_1, J_2, \ldots, J_{j-1}$ in a schedule like σ and the set $\{J_{k+1}, \ldots, J_{j-1}\}$ yields the last batch in the current schedule σ. For initializing set $F(0) = 0$. On the other hand imagine the job which has to be assigned on the machine is the jth job that this machine processes. Also in the rth group ($r = 1, 2$), it is the J_{ji_r} turn to be assigned and $i_r - 1$ jobs in this group have been assigned so far (J_{ji_r} means the ith job of the rth group when it is assigned as the jth job on the machine). Therefore, there should be a set like the form $A = \{J_{ji_1}, J_{ji_2}\}$ in which each member has the possibility to be scheduled on the machine. The cost (completion time) for assigning J_{ji_r} as the jth job on the machine is $F(J_{ji_r})$. Of the jobs in A, one job is assigned which has the lowest value of F. Hence one of the jobs in A which attains F_{min} in (4) is deserved to be scheduled.

$$F_{min} = \mathrm{Min}\{F(J_{ji_1}), F(J_{ji_2})\} \tag{4}$$

In order to calculate the value of $F(J_j i_r)(r = 1, 2)$ we have the following DP algorithm:

Table 5
Results for the problem $1|p\text{-}batch, b \leqslant n, IG, mul\text{-}cust|F(C_{max}, L_{max})$ and 100-job instances.

Type	θ	Capacity of batches											
		5				10				15			
		CPU time		Objective value		CPU time		Objective value		CPU time		Objective value	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
$n = 100$													
I	0.1	0.001	0.016	196.1	207.4	0.002	0.016	147.1	158.1	0.002	0.016	127.3	142.1
	0.2	0.002	0.016	291.6	317.8	0.002	0.016	179.5	209.2	0.002	0.016	45.1	125.2
	0.3	0.002	0.016	372.9	420.7	0.001	0.016	21.1	78.9	0.002	0.016	−184.1	40.5
	0.4	0.002	0.016	442.1	534.4	0.001	0.016	−258.2	−30.8	0.002	0.016	−382.6	−235.6
	0.5	0.002	0.016	291.9	533.5	0.002	0.016	−541.1	−362.5	0.001	0.015	−647.1	−434.1
	0.6	0.003	0.016	23.8	139.8	0.002	0.016	−735.4	−471.1	0.002	0.016	−953.1	−675.4
	0.7	0.002	0.016	−257.8	−29.6	0.002	0.016	−977.5	−569.9	0.002	0.016	−1221.2	−829.6
	0.8	0.002	0.016	−360.8	−159.1	0.002	0.016	−1282.4	−1055.2	0.002	0.016	−1407.1	−1032.8
	0.9	0.001	0.016	−573.1	−148.7	0.002	0.016	−1544.2	−1125.1	0.002	0.016	−1657.2	−1076.6
II	0.1	0.002	0.016	195.1	205.6	0.001	0.016	143.1	157.8	0.002	0.016	122.8	136.2
	0.2	0.002	0.016	288.4	319.6	0.002	0.016	75.8	172.6	0.002	0.016	−122.7	−3.4
	0.3	0.003	0.016	337.2	411.6	0.001	0.016	−255.2	−35.1	0.002	0.016	−446.1	−256.2
	0.4	0.002	0.016	138.2	278.4	0.001	0.015	−656.6	−445.8	0.002	0.016	−775.4	−539.1
	0.5	0.002	0.016	−265.6	−25.5	0.001	0.016	−987.3	−562.5	0.001	0.016	−1088.6	−682.5
	0.6	0.003	0.016	−598.7	−361.4	0.002	0.016	−1498.1	−1085.4	0.001	0.015	−1477.8	−955.1
	0.7	0.002	0.016	−909.9	−604.4	0.001	0.016	−1774.6	−1386.4	0.002	0.016	−1846.6	−1356.6
	0.8	0.002	0.016	−1317.3	−994.2	0.001	0.016	−1986.9	−1615.8	0.002	0.016	−2142.4	−1545.8
	0.9	0.002	0.016	−1776.7	−1179.6	0.001	0.016	−2566.3	−2231.6	0.002	0.016	−2422.9	−1615.1
III	0.1	0.003	0.016	190.3	204.1	0.002	0.016	139.9	154.6	0	0	123.1	135.4
	0.2	0.001	0.015	287.1	309.4	0.001	0.016	72.9	200.1	0.002	0.016	−76.7	75.4
	0.3	0.003	0.016	317.1	403.3	0.002	0.016	−287.6	−67.9	0.002	0.016	−476.9	−251.3
	0.4	0.003	0.016	85.1	292.8	0.001	0.015	−669.5	−469.8	0.002	0.016	−709.2	−474.1
	0.5	0.002	0.016	−278.6	−76.5	0.001	0.016	−947.8	−756.1	0.001	0.016	−1190.5	−828.1
	0.6	0.003	0.016	−709.2	−462.1	0.001	0.016	−1297.3	−1133.2	0.002	0.016	−1529.6	−1053.2
	0.7	0.002	0.016	−1002.8	−597.7	0.001	0.016	−1721.9	−1511.1	0.002	0.016	−1863.1	−1392.7
	0.8	0.002	0.016	−1257.1	−794.1	0.001	0.015	−2112.7	−1717.1	0.001	0.016	−2339.1	−1814.8
	0.9	0.002	0.016	−1607.8	−1178.5	0.002	0.016	−2515.2	−1774.6	0.002	0.016	−2456.7	−1801.9

$$F(J_{j_{i_r}}) = \mathrm{Min}\{F(J_k) + p_{\max}\}$$
$$s.t. \begin{cases} 0 \leqslant k < j \\ \sum s_t \leqslant b, F(J_k) + p_{\max} \leqslant Min\{d_t + L\}(k+1 \leqslant t \leqslant j-1 \cup \{i_r\}) \end{cases} \quad (5)$$

where $p_{\max}$ is the processing time of the job J_t and d_t is $+\infty$ for jobs of the class C_1. Also s_t represents job size which is equal to one for all jobs. After calculating the value of F for jobs in A, we find the job which attains F_{min}. After that we set:

$$F(J_j) = F_{\min} \quad (6)$$

Finally the optimal F is equal to $F(J)$ and the optimal schedule is found by backtracking. For more details suppose $\bar{k}_j$ is a value of k which attains the minimum value of $F(J_k) + p_{i_r}$ in (5). The optimal schedule is gained by considering $\{J_{k_{n_1}+1}, \ldots, J_{n_1}\}$ as the last batch and $\{J_{\bar{k}_{i+1}}, \ldots, J_{\bar{k}_{n_1}}\}$ as the second last batch and by this way the rest of batches can be found. When the final schedule is obtained, the value of $F(J)$ represents $C_{\max}$ and regarding to that L or allowed $L_{\max}$ which is assumed at the beginning of this method, we have the first Pareto optimal point or $(C^{(1)}_{\max}, L^{(1)}_{\max})$. To have the other Pareto optimal points we decrease the current L by one unit and repeat the above process and calculate $C_{\max}$, thus the Pareto optimal points are generated one by one. This process is continued until decreasing in L does not result in a feasible schedule. Because of integral data there is no sense to have a schedule whose $L_{\max}$ value is between $L_{\max}(\sigma)$ and $L_{\max}(\sigma - 1)$. The above process which finds the Pareto optimal set is given in the following algorithm.

Step 1: Suppose the pair $(C^{(1)}_{\max}, L^{(1)}_{\max})$ is available and σ_1 is its corresponding related schedule.
Step 2: When the ith Pareto optimal pair or $(C^{(i)}_{\max}, L^{(i)}_{\max})$ is gained, set $L = L^{(i)}_{\max} - 1$ and solve the problem $1|p\text{-}batch, b \leqslant n, CG, mul\text{-}cust, L_{\max} \leqslant L|C_{\max}$ using the given dynamic programming algorithm. Go to step 3.
Step 3: If the current solution is infeasible, then return all Pareto optimal schedules $\sigma_1, \sigma_2, \ldots, \sigma_i$ and stop. Otherwise the schedule σ_{i+1} is the (i + 1)th Pareto optimal schedule where $L^{(i+1)}_{\max} = L_{\max}(\sigma_i) - 1$. Therefore, the pair $(C^{(i+1)}_{\max}, L^{(i+1)}_{\max})$ is the $(i + 1)$th Pareto optimal point. Set $i = i + 1$ and go to step 2.

In order to start the above algorithm we have to know the value of L or allowed $L^{(1)}_{\max}$. The following description provides the way of finding the starting $L_{\max}$ in the first Pareto optimal point.

Table 6
Results for the problem $1|p\text{-}batch, b \leqslant n, IG, mul\text{-}cust|F(C_{\max}, L_{\max})$ and 150-job instances.

Type	θ	Capacity of batches											
		5				10				15			
		CPU time		Objective value		CPU time		Objective value		CPU time		Objective value	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
$n = 150$													
I	0.1	0.006	0.016	249.1	267.3	0.003	0.016	174.3	185.4	0.003	0.016	142.6	150.7
	0.2	0.007	0.016	393.8	441.1	0.003	0.016	213.6	259.8	0.003	0.016	−10.2	140.2
	0.3	0.008	0.016	513.8	559.6	0.002	0.016	−15.2	167.8	0.003	0.016	−404.1	−200.4
	0.4	0.008	0.016	598.2	682.4	0.003	0.016	−449.4	−185.4	0.004	0.016	−873.1	−523.4
	0.5	0.006	0.016	386.7	605.5	0.003	0.016	−846.5	−675.5	0.003	0.016	−1326.1	−1077.5
	0.6	0.006	0.016	5.9	132.4	0.004	0.016	−1266.3	−958.6	0.002	0.016	−1701.1	−1397.4
	0.7	0.007	0.016	−323.6	128.1	0.004	0.016	−1730.1	−1388.4	0.003	0.016	−2056.2	−1554.9
	0.8	0.006	0.016	−815.4	−567.6	0.003	0.016	−2093.8	−1829.1	0.003	0.016	−2640.9	−2349.2
	0.9	0.006	0.016	−984.2	−684.1	0.003	0.016	−2538.5	−1645.9	0.004	0.016	−2906.2	−2187.4
II	0.1	0.007	0.016	240.7	258.2	0.002	0.016	168.1	174.2	0.003	0.016	142.8	155.6
	0.2	0.007	0.016	352.9	406.4	0.003	0.016	60.7	190.6	0.003	0.016	−332.7	−196.1
	0.3	0.007	0.016	470.6	581.8	0.002	0.016	−524.2	−340.4	0.002	0.016	−994.6	−708.7
	0.4	0.006	0.016	63.8	233.4	0.003	0.016	−1096.6	−736.6	0.003	0.016	−1538.1	−1117.2
	0.5	0.009	0.016	−424.3	−142.1	0.003	0.016	−1642.6	−1230.5	0.002	0.016	−2156.2	−1882.5
	0.6	0.006	0.016	−991.1	−631.8	0.003	0.016	−2348.1	−1989.1	0.003	0.016	−2574.2	−1875.6
	0.7	0.007	0.016	−1567.3	−1238.6	0.003	0.016	−2784.2	−2204.9	0.004	0.016	−3103.6	−2487.1
	0.8	0.005	0.016	−2068.6	−1791.8	0.003	0.016	−3544.4	−2946.8	0.002	0.016	−3699.5	−2984.1
	0.9	0.006	0.016	−2692.9	−2102.6	0.003	0.016	−4228.8	−3468.9	0.004	0.016	−4413.2	−3828.1
III	0.1	0.007	0.016	245.2	260.6	0.004	0.016	162.1	182.6	0.004	0.016	127.3	149.8
	0.2	0.005	0.016	383.8	415.8	0.003	0.016	59.9	216.8	0.003	0.016	−317.2	−183.1
	0.3	0.006	0.016	455.7	537.7	0.003	0.016	−511.5	−368.3	0.003	0.016	−864.6	−628.1
	0.4	0.007	0.016	68.1	213.1	0.003	0.016	−1097.7	−805.6	0.003	0.016	−1428.1	−988.4
	0.5	0.007	0.016	−407.5	−217.1	0.002	0.016	−1596.7	−1369.1	0.003	0.016	−2144.3	−1796.1
	0.6	0.007	0.016	−963.1	−630.1	0.004	0.016	−2248.1	−1925.1	0.003	0.016	−2775.4	−2270.8
	0.7	0.006	0.016	−1587.2	−1247.8	0.002	0.015	−2924.4	−2521.3	0.003	0.016	−3213.7	−2608.8
	0.8	0.007	0.016	−2023.8	−1618.2	0.004	0.016	−3412.1	−2712.6	0.003	0.016	−3993.6	−3029.6
	0.9	0.006	0.016	−2752.1	−2402.1	0.003	0.016	−4138.2	−3713.4	0.002	0.016	−4487.5	−3817.3

Consider J_j as the job with the smallest due date among all jobs and consider it in the batch B_1 (this batch only contains the job J_j). Similarly we distribute the other jobs in batches such that each batch contains only one job. The sequence order of these jobs is not important. Let the batch B_1 be scheduled at the end after all batches. Thus the job J_j attains the maximum lateness among all jobs. Consider this L_{max} as the maximum lateness of the current schedule and name it L^*_{max}. It is clear that the C_{max} of this schedule or C^*_{max} is very large too, thus (C^*_{max}, L^*_{max}) may be dominated by another pair and clearly this pair is not a Pareto optimal point. If we consider L^*_{max} as L in (5), the restriction $F(J_k) + p_{i_r} \leqslant \mathrm{Min}\{d_t + L\}(k+1 \leqslant t \leqslant j-1 \cup \{i_r\})$ is always satisfied. On the other hand since L^*_{max} is very large, we are sure there is L^{**}_{max} where $L^{**}_{max} \ll L^*_{max}$ such that all of the schedules whose maximum lateness values are between L^*_{max} and L^{**}_{max} have similar C_{max} or C^*_{max} (because for these schedules the above mentioned restriction in (5) is always satisfied) and consequently all of them will be dominated by the pair $(L^{**}_{max}, C^{**}_{max})$. Now for finding L^{**}_{max}, we solve the DP algorithm for $L = L^*_{max}$ and consider the largest occurred value of $F(J_k) + p_{i_r} - d_t$ and set $L^{**}_{max} = L^{(1)}_{max} = F(J_k) + p_{i_r} - d_t$. Hence by obtaining C_{max} or $C^{(1)}_{max}$, the first Pareto optimal point or $(C^{(1)}_{max}, L^{(1)}_{max})$ will be resulted. In other words we gained the needed value of L_{max} or L for starting the algorithm.

The subsequent section describes the results obtained from UIOO, BIOO and BCOO on a random set of instances. The following study helps to observe the ability of the three algorithms in solving their related problems together with the required running times.

4. Computational experiments

In this section a set of samples was used to test the performance of UIOO, BIOO and BCOO. Since the three proposed problems have not been considered in the literature so far, there is no benchmark algorithm to be compared with our optimal algorithms. Random instances were generated to study the effect of job numbers, due dates, batch sizes and scaling factors when they change in problems. Also the required running times of performing the algorithms are calculated. Processing times of jobs were generated in the discrete uniform interval [0,100] for all test instances. To study the effect of Tight, Moderate and Loose due dates, three sampling intervals [0,0.75P], [0,P] and [0.25P,P] were considered, respectively where P is the sum of processing times. Without losing the generality, we consider the same job numbers for the C_1 and C_2 or $n_1 = n_2 = n$. For each problem we randomly generated 20 instances and provide the average values over 20 instances in tables.

Since we have two groups of jobs, we generate two sample sets for each group. For the problems $1|p\text{-}batch, b \leqslant n, CG, mul\text{-}cust|\ F(C_{max}, L_{max})$ and $1|p\text{-}batch, b \leqslant n, IG, mul\text{-}cust|F(C_{max}, L_{max})$ the batch capacities were assume to be 5, 10 and 15.

In this study, all experimental tests were conducted on a personal computer with Pentium IV/2 512 RAM. Tables 1–3 represent the obtained results when UIOO is employed to solve the problem $1|p\text{-}batch, b > n, IG, mul\text{-}cust|F(C_{max}, L_{max})$ with 50, 100 and 150 jobs, respectively. The first column in all tables shows the type of due date such that type 1, 2 and 3 represent Tight, Moderate and Loose due dates, respectively. The second column implies kind of θ which is applied in $Z = \theta L_{max} + (1-\theta C_{max})$ and is equal to 0.1, 0.2, ..., 0.9. We show the average and maximum objective values over 20 instances by Objective Value in tables in column 5 and 6, respectively. Column 3 and 4 show the average and maximum time obtained during performing 20 instances for each case. The time is calculated based on second and the results show that the final solution in all instances is obtained in less than 0.047 s. This is a reasonable time despite the fact that the degree of complexity is more pronounced as the job size gets larger.

Table 7
Results for the problem $1|p\text{-}batch, b \leqslant n, CG, mul\text{-}cust|F(C_{max}, L_{max})$.

			$n = 50$			$n = 100$			$n = 150$		
			Type I	II	III	I	II	III	I	II	III
Capacity of batches											
5	OVCmax	Mean	870.8	877.4	886.7	1730.1	1741.7	1756.1	2565.9	2573.2	2564.1
		Max	952.8	958.3	975.1	1857.3	1824.8	1868.5	2732.2	2668.5	2693.1
	OVLmax	Mean	125.1	127.1	116.2	225.1	254.9	249.1	333.1	332.9	313.2
		Max	205.1	207.3	188.6	370.3	304.4	331.7	416.9	424.2	446.1
	CPU time	Mean	0.016	0.018	0.021	0.069	0.082	0.061	0.102	0.099	0.106
		Max	0.031	0.032	0.047	0.109	0.094	0.094	0.141	0.141	0.156
10	OVCmax	Mean	516.1	527.4	516.8	982.1	986.9	974.2	1414.9	1407.1	1440.2
		Max	576.7	572.8	567.3	1081.4	1033.7	1031.4	1457.8	1493.1	1522.2
	OVLmax	Mean	83.9	53.9	56.3	107.7	141.4	123.9	188.2	170.2	199.9
		Max	152.8	149.2	131.3	192.6	198.2	172.8	297.1	233.1	268.6
	CPU time	Mean	0.038	0.043	0.052	0.102	0.119	0.134	0.195	0.195	0.201
		Max	0.062	0.063	0.109	0.125	0.172	0.172	0.234	0.251	0.266
15	OVCmax	Mean	374.3	373.4	368.1	693.4	694.1	683.1	1008.4	1001.5	991.4
		Max	404.1	394.7	408.7	734.1	746.9	754.9	1087.9	1044.8	1038.8
	OVLmax	Mean	55.6	50.4	73.7	114.2	101.6	107.9	150.4	125.7	123.5
		Max	102.6	100.3	104.4	165.4	145.2	157.8	193.1	202.1	174.2
	CPU time	Mean	0.055	0.073	0.075	0.145	0.186	0.153	0.291	0.317	0.294
		Max	0.078	0.094	0.109	0.172	0.219	0.187	0.375	0.375	0.375

Tables 4–6 provide the results of solving the problem 1| *p-batch*, $b \leqslant n$, *IG*, *mul-cust*|$F(C_{max}, L_{max})$ for $n = 50$, 100 and 150 when BIOO is applied. The objective function is the same as the previous problem. We consider the average and maximum values of the objective function obtained in all 20 instances and report the results in the Objective Value column. Since batches are bounded in this problem we consider batch sizes as 5, 10 and 15. As it is seen from the tables, the number of jobs does not have a considering impact on the needed running time of the algorithm.

We show the results for the problem 1|*p-batch*, $b \leqslant n$, *CG*, *mul-cust*|$F(C_{max}, L_{max})$ when $n = 50$, 100 and 150 in Table 7. Since jobs of the C_2 group have the same processing time, we generate n numbers in [1,100] and without lose of generality we consider their average as the processing time of the jobs in C_2. As it was mentioned before, the BCOO method finally finds the Pareto optimal set whose members are like the form $(C^{(i)}_{max}, L^{(i)}_{max})$. To show the final value as a judgment tool in Table 7, first we obtain the average of $C^{(i)}_{max}$ and name it AveCmax. Similarly we have AveLmax by obtaining the average of $L^{(i)}_{max}$. Then we compute OVCmax $|_{mean}$ and OVCmax $|_{max}$ by finding the average and maximum AveCmax through 20 generated instances. OVLmax$|_{mean}$ and OVLmax$|_{max}$ are gained in a similar way by considering AveLmax in all instances. We also report the spent time for running the algorithm. As it is seen, the needed time is 0.375 in the worst case. Thus the BCOO algorithm is able to process a large number of jobs in batches in a very short time.

5. Conclusion

Our paper assumes the parallel batch processing model with makespan and maximum lateness criteria in a multi-agent model. A multi-agent model considers different competing customers who pursue different objectives and each customer represents a job group. A customer has to select his jobs from his own group so he has a direct role in minimizing his related objective function. Regarding to this model, three problems are studied which are differentiated in batch capacity and the compatibility among job groups. We consider two main branches when these groups can be compatible or incompatible. In the case when groups are compatible, jobs from different groups can be inserted in a same batch while it is impossible when job groups are incompatible. By considering batch capacity in each category, we study three diverse problems and proposed optimal methods for them. In one case that groups are incompatible and batches are bounded we note our method as a heuristic when processing times of the group with maximum lateness objective are different. In other words the proposed method for this case is optimal when these processing times are identical. As the computational experiment shows the behavior of the three proposed algorithms performs well in terms of the running time for small, medium and large sized problems.

References

[1] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, A. Pacifici, Nondominated schedules for a job-shop with two competing users, Comput. Math. Org. Theor. 6 (2000) 191–217.
[2] A. Agnetis, P.B. Mirchandani, D. Pacciarelli, A. Pacifici, Scheduling problems with two competing agents, Oper. Res. 52 (2) (2004) 229–242..
[3] Kenneth R. Baker, J. Cole Smith, A multi-criterion model for machine scheduling, J. Scheduling 6 (2003) 7–16.
[4] He Cheng, Yixun Lin, Jinjiang Yuan, Bicriteria scheduling on a batching machine to minimize maximum lateness and makespan, Theor. Comput. Sci. 381 (2007) 234–240.
[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, Reading, Masachusetts, 1989.
[6] D. Bertsekas, Dynamic Programming: Deterministic and Stochastic Models, Prentice-Hall, NJ, 1976 (Chapter 1).
[7] Peter Brucker, Andrei Gladky, Han Hoogeveen, Mikhail Y. Kovalyov, Chris N. Pots, Thomas Tautenhahn, Steef L. Van De Velde, Scheduling a batching machine, J. Scheduling 1 (1998) 31–54.